Title	数式処理システムの数学教育への活用とミドルレギュレータの開発 (数式処理における理論と応用の研究)
Author(s)	出口, 博章; 高橋, 正
Citation	数理解析研究所講究録 (2000), 1138: 201-210
Issue Date	2000-04
URL	http://hdl.handle.net/2433/63815
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録
1138巻 2000年 201-210

数式処理システムの数学教育への活用とミドルレギュレータの開発

神戸大学大学院自然科学研究科　出口　博章(Hiroaki Deguchi) *
神戸大学発達科学部　高橋　正(Tadashi Takahashi) †

概　要

近年、数学教育において、数式処理システムの利用が求められている。しかしながら、現状では、個々の教師の試みとしての活動に近いと言える。その理由としては、システムの購入経費，インストール及びメンテナンスの負担等が一因である。

それに対して、新設教科「情報科」を設ける等、学校教育の現場では、インターネットの普及に伴い、ネットワーク環境が整いつつある。インターネットの利用として第一に普及しているものは、e-mail と www である。この e-mail と www を数式処理システムのユーザインターフェースとして用い、数式処理システムを www ブラウザのアプリケーションの一つとすることで、数式処理システムの学校教育での利用を、敷居の低い、使い易いものにしたい。

上記の目的のために、 e-mail と www を用いたミドルレギュレータの開発を行った。さらに、ミドルレギュレータは、数式処理システムに統一的なインタフェースを与えることを目的として開発された。

1　数式処理システムの教育利用の在り方

教育の現場で、数式処理システムを利用することによって、次のようなことが新たに可能となる。

（1）手計算ではできなかった複雑な問題を、例題や演習問題として取り上げることが可能になる。

（2）グラフ表示機能を利用して、数学をより視覚的・動的に理解することができる。

（3）新しい概念の習得に必要な計算技術を十分に習得していない生徒も、生徒自身が数式処理システムを利用することにより、新しい概念を習得できる。

（4）数式処理システムを通じて、大量の例を作ることができる。

*ydeg@kobe-u.ac.jp
†takahasi@kobe-u.ac.jp

これらのことによって、

計算機実験→観察→定理の予想→証明
あるいは、
予想→計算機実験→観察→証明

という形の数学の修得が可能になる。従って、数式処理システムは、数学教育の教授法をより多様にできる強力な道具であり、数式処理システムが家庭や企業に普及して行けば、広い意味で、数学教育を大きく変えることになるであろう。

2 ミドルレギュレータの目的と位置付け

ミドルレギュレータは、数式処理システムに統一的なインタフェースを与えることを目的として開発された。人間の思考を助けるツールとして数式処理システムをとらえると、本来数式処理システムによって得られる利益以外の余分な労力を省きたいというのが、究極的な目的である。

2.1 ミドルレギュレータを作成するに至った動機

コンピュータ利用を伴わない数式処理の場面、例えばノートに数式を書きながら考える場面とコンピュータ利用を伴う数式処理の場面を比較した場合、後者においては、数式処理システムを使うための技能がが必要とされる。なぜならば、数式の意味をコンピュータに理解させるための記述方法は、ノートに数式を書くほどの自由度がなく、x^3 を表わすには `x^3` と記述するような知識が要求される。さらに、何らかの操作の指示、例えば「この数式の展開はどのように実行させるのか」という知識も必要とされる。つまりユーザの視点から大きく分けると、

- (数式処理システムを使う) 以前に (ノートなどに) 書いていた数式を数式処理システムが理解できる表現方法に翻訳するための知識
- (数式処理システムを使う) 以前に (ノートを使用したり暗算したりしていた) 演算を数式処理システムに実行させるための知識

以上の二種類の知識を習得して初めて使用できるのが、現在の数式処理システムである。

コンピュータ利用による数式処理を行なう際の思考の流れに着目すると、表現したい内容を、数式を利用して抽象化するというステップを経て、その結果をさらに翻訳するという第二のステップが必要となる。

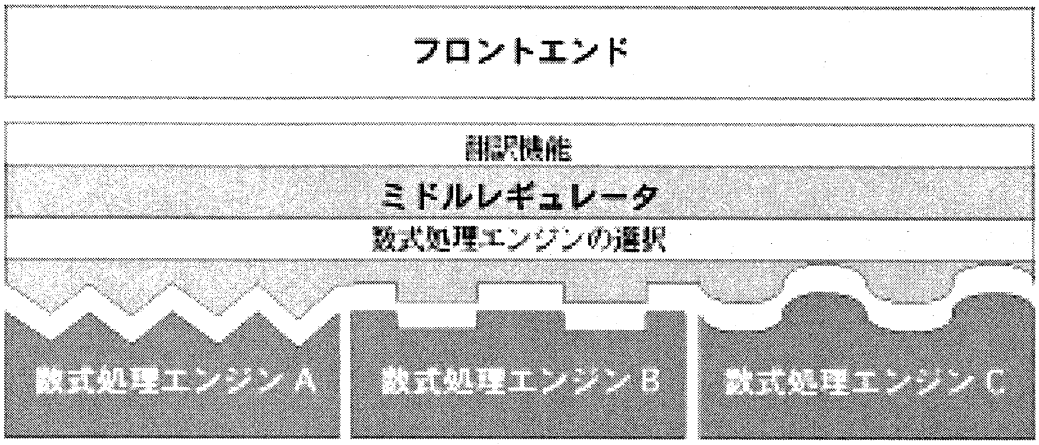

図 1: ミドルレギュレータ (Middle-regulator) = 中間で調整するオブジェクト

このような場面においては、情報の発信者か受信者のどちらかがその余分なプロセスを負担しなければならないが、情報伝達の遅延による支障が少ないのならば両者の間に翻訳専門の第三者を付加すれば、両者の思考プロセスの負担が軽減されることが期待される。

2.2 ミドルレギュレータの位置付け

数式処理システムとユーザの間に翻訳者を置き、そのオブジェクトをミドルレギュレータと名付けた (図 1)。レギュレータは「調整するオブジェクト」という意味であり、ミドルには「ユーザ・インタフェースと数式処理エンジンの中間」という意味と、将来的にはユーザ・インタフェースか数式処理エンジンのどちらか、あるいはその両方が、ミドルレギュレータの持つ機能を吸収することが望ましいという「(最終生成物をにらんでの) 中間生成物」という意味をもたせている。図 1 にあるように、複数の数式処理システムに対応することも目標とする。また、図 1 に「数式処理エンジン」としているのは、数式処理を行なう部分を意識してエンジンと表現している。各数式処理上部の境界の形状の差異は数式処理エンジン固有のアクセス方法 (オブジェクトに対する通信) の違いを表現している。ミドルレギュレータ上部で境界線が平らになっていることは、ミドルレギュレータが各数式処理エンジンの差異を吸収するということを意味する。

3 現在のミドルレギュレータ

今回新たに付加されたのは「通信と演算を非同期で行なう機能」である。この機能によって数式処理エンジンでの演算の終了を待たずに、ユーザの端末を解放することが可能になった。

また、現在のミドルレギュレータはインタフェースを備えたオブジェクト群として開発されているため、将来的にミドルレギュレータとしての機能を更新する事が以前よりも容易になった。

3.1 通信と演算を非同期で行なう機能

現在作成されているものは、ウェブブラウザから利用できるオブジェクトとして設計されている。このことはインタフェースはウェブブラウザに対してのみであるという限定を意味するものではない。ユーザと数式処理システムのインタフェースとして機能するために、内部に翻訳機能を持つことが必須条件となる。現実的には紙に鉛筆で自由に数式を記述することに匹敵するユーザ・インタフェースがまだ存在しないため、複数の数式処理システムを同じような操作で扱えるようにするという統一的なインタフェースを備えるという事が現在の妥協点となる。

今回新たに追加した機能は「ユーザと数式処理システム間の接続」と「数式処理システムの稼働」を非同期で処理する機能である。ユーザの使用するクライアントマシンが接続されている時間と、サーバの数式処理システムが稼動している時間が一致するのではなく、ユーザが明示的に終了の合図を送るまではクライアントマシンからの接続が中断されてもサーバは待機することになる。

この機能の利点は、計算時間が長くユーザがかなり待たされるような場面において、その間クライアントマシンの動作を拘束しないというところにある。思考のためのツールが停止した状態において、思考の中断がなくなり、ユーザはクライアントマシンを利用して別の作業をすることが可能になる。

以上のような機能を備えるために、ミドルレギュレータ内部に Math サーバとエントリサーバというオブジェクトを新たに付加し、ウェブブラウザとのインタフェースとなる CGI 部とあわせて、三つのオブジェクトの集合とした (図 2)。ユーザが式を送信してから結果を受取るまでの標準的な流れは次のようになる。

(1) まず、ブラウザ上のインプットフォームに入力された式が HTTP のフォームとしてウェブサーバに送られる。

(2) ウェブサーバは送られてきた URL から CGI 部を起動し、送られてきたフォームの文字数を環境変数として、またフォームの内容は CGI 部の標準入力に対して送る。

(3) CGI 部では送られてきたフォームの文字数を環境変数から取出し、フォームの内容を標準入力から得る。そして得られた文字列をパターンマッチ表と照合し翻訳の必要があれば翻訳する。またブラウザ端末の IP アドレスを環境変数から取得し、それをユーザ ID としてエントリサーバに通知する。

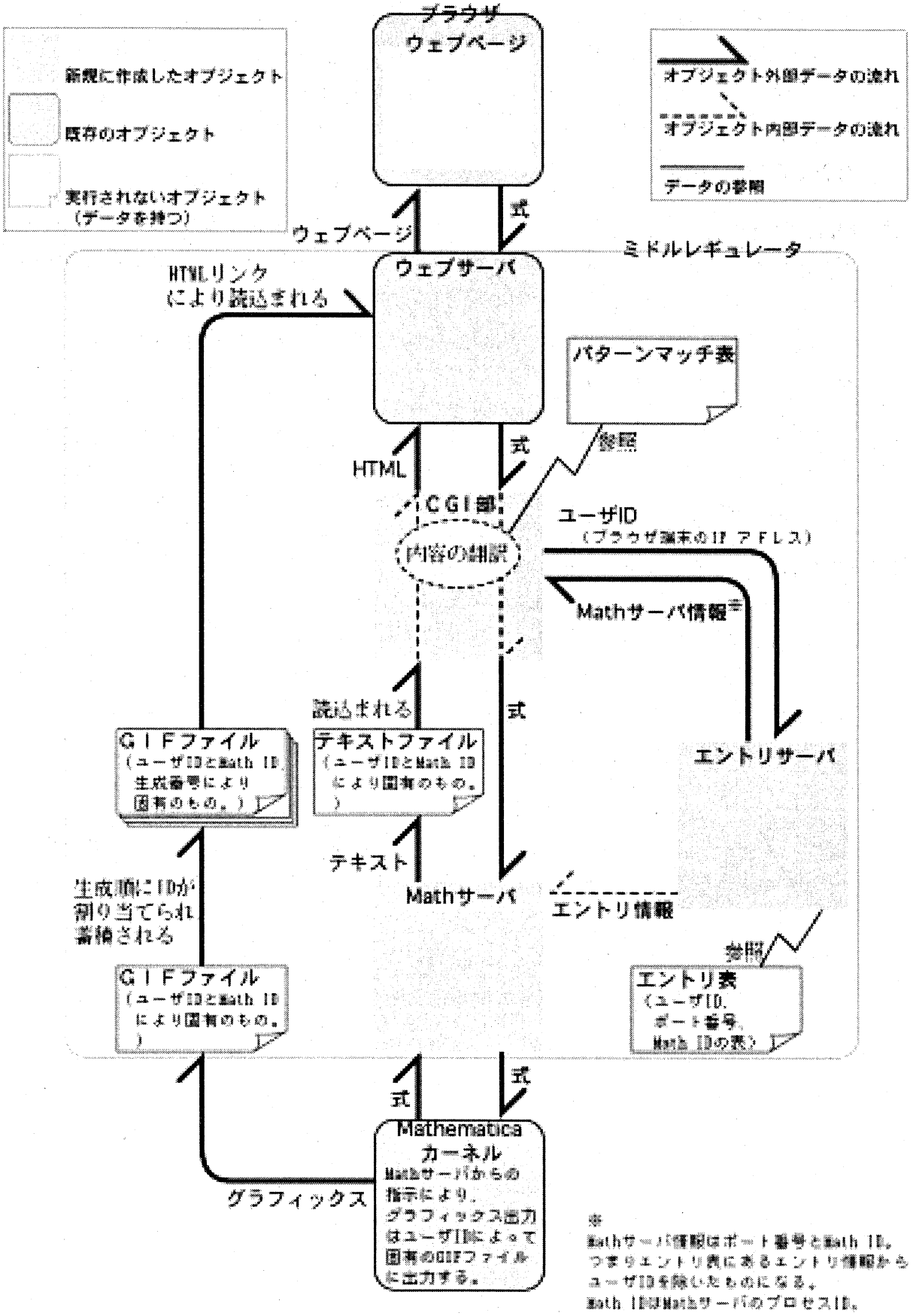
ブラウザ
ウェブページ
新規に作成したオブジェクト
既存のオブジェクト
実行されないオブジェクト
（データを持つ）
オブジェクト外部データの流れ
オブジェクト内部データの流れ
データの参照
ウェブページ
式
ミドルレギュレータ
HTMLリンク
により読込まれる
ウェブサーバ
パターンマッチ表
参照
HTML
式
CGI部
内容の翻訳
ユーザID
（ブラウザ端末のIP アドレス）
Mathサーバ情報
読込まれる
式
GIFファイル
（ユーザIDとMath ID
生成番号により
固有のもの。）
テキストファイル
（ユーザIDとMath ID
により固有のもの。
）
エントリサーバ
テキスト
Mathサーバ
エントリ情報
生成順にIDが
割り当てられ
書換される
参照
GIFファイル
（ユーザIDとMath ID
により固有のもの。
）
エントリ表
（ユーザID、
ポート番号、
Math IDの表）
式
式
Mathematica
カーネル
Mathサーバからの
指示により、
グラフィックス出力
はユーザIDによって
固有のGIFファイル
に出力する。
グラフィックス
※
Mathサーバ情報はポート番号とMath ID。
つまりエントリ表にあるエントリ情報から
ユーザIDを除いたものになる。
Math IDはMathサーバのプロセスID。

図 2: データの流れ

(4) エントリサーバはCGI部からの通信があるとその内容をユーザIDと判断しエントリ表と照合する。受取ったユーザIDがエントリ表にあれば既にMathサーバが起動中であることを意味するのでMathサーバ情報をCGI部に通知する。受取ったユーザIDがエントリ表に存在しなかった場合はMathサーバを起動し、Mathサーバ情報をCGI部に通知する。

(5) Mathサーバはエントリサーバによって起動されるが、起動の際に、エントリ情報をエントリサーバから引き継いでいる。そのエントリ情報によって、各ユーザID毎に結果のファイルを格納するスペースを作成し、Mathematicaのカーネルを呼び出す。そしてグラフィックス出力をユーザID毎に別のGIFファイルに格納するための通信を行ない、CGI部からの通信を待つ。

(6) CGI部にとってはMathサーバが今回新たに起動されたかものか以前から起動していたものかに関わらず、エントリサーバからはMathサーバ情報が通知されるので、その情報を使用してMathサーバに式を送信する。

(7) CGI部から式を受取ったMathサーバはその式をカーネルに送り、その式に対するカーネルの式を受取るとテキストファイルとして書出す。この時、カーネルから受取った式がグラフィックス出力を含むかどうかをチェックし、グラフィックス出力を含む場合は出力されたグラフィックスファイルの名前に生成番号を付加し、次のグラフィックスファイル出力があった場合に上書きされないようにする。同時に生成番号を含むグラフィックスファイル固有の価をテキストファイルに書出す。

(8) CGI部ではMathサーバに式を送り終わった時点で、結果が書出されているテキストファイルを読込む。その段階ではまだカーネルが演算中の場合は、その演算の直前までの出力結果を取得することになる。読込まれたテキストファイルの内容にHTMLタグを付加しウェブサーバに送信する。テキストファイルにグラフィックスファイル出力を示す価が書込まれていた場合には、そのグラフィックスファイルに対するHTMLリンクに翻訳する。

(9) ウェブサーバはCGI部から送られてきたHTMLテキストをブラウザに送信する。HTMLテキストの中にグラフィックスファイルへのHTMLリンクが含まれる場合には、ウェブサーバにグラフィックスの送信を要求し、取得後にHTMLによって設定されたフォーマットに基づいてウェブページを出力する。

出力結果が一度ファイルに落とされているのは、数式処理システムが演算中であっても、現在までの出力結果を、クライアントマシンに送付するためである。また、このことによって初回の接続時以外はウェブページにアクセスした時に、初回の接続から現在までの出力結果が表示されるという事が実現されている。「二つの入力 `x`, `y` に対して、"TRUE"か"FALSE"、

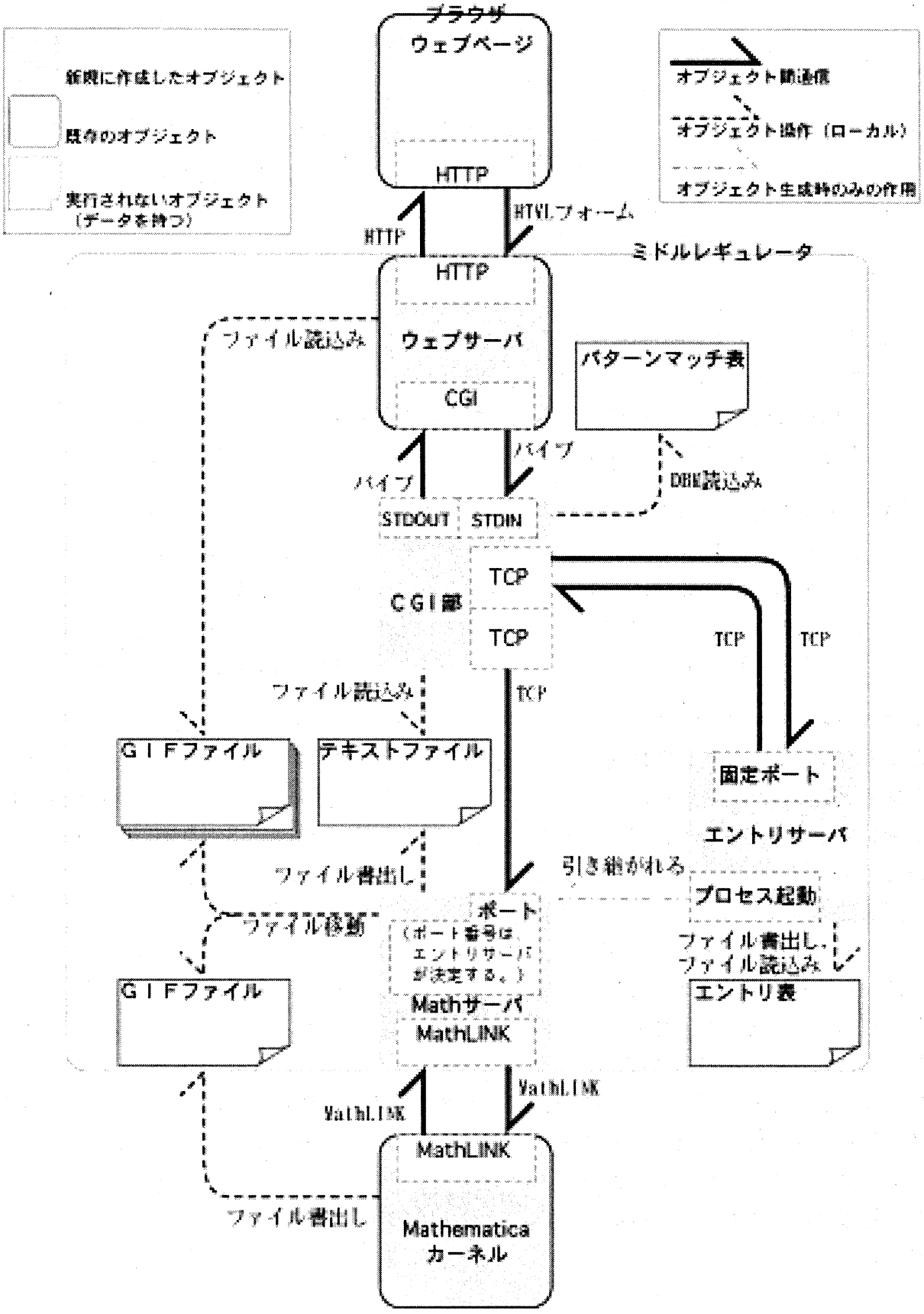

図 3: 各オブジェクトのインタフェース

または“判断不能”を返し、要求があればその理由を返す」オブジェクトを仮定すると、これは簡潔なインタフェースを備えているといえる。しかし、「二つの入力 x, y に対して、“別のオブジェクトへの参照”か“TRUE”、“FALSE”、“判断不能”を返し要求があれば、その理由を返す。“別のオブジェクトへの参照”を返した場合には、さらにその戻り値を受け付けて“TRUE”、“FALSE”、“判断不能”を返し要求があればその理由を返す」オブジェクトであるならば、簡潔であるとは言い難い。

ミドルレギュレータにおいて「簡潔なインタフェース」は次の二つの指針により実現している。

- オブジェクトの間で通信されるデータの意味の簡潔さ
  図 2 中の各矢印はオブジェクト間を流れるデータの方向を表わしている。矢印の傍らにその経路を流れるデータの説明が付されている。それぞれの矢印の持つデータの意味が明確であり、場合によって異なる意味を持つということはない。

- 各オブジェクト間においての経路の簡潔さ
  図 3 中の各矢印はオブジェクトの持つインタフェースの作用する方向を表わしている。実線の矢印については開始する点と終了する点にインタフェースの説明が付されている。矢印によって結ばれているオブジェクトを「隣接するオブジェクト」と呼ぶならば、隣接する任意のオブジェクトの間には矢印が最大で 2 つである。

ミドルレギュレータ全体を 1 つのオブジェクトと考えた場合には Mathematica カーネルとミドルレギュレータの間に矢印が 3 つ存在することになるが、これは Mathematica のグラフィックスファイル出力を利用しているためである。厳密には UNIX 版の Mathematica においてはカーネルからグラフィックスファイル出力のための別プロセスが呼び出されているため、グラフィックスファイル出力用の別オブジェクトが存在することになる。

図 4 は図 3 におけるオブジェクトの関係をさらに簡潔に表現したものである。各オブジェクトは大きな円として表現されており、そのオブジェクトが他のオブジェクトに対して持っているインタフェースが小さな円で表現されている。オブジェクト操作も「〜しろ」という内容の通信であるため、1 つの矢印はあるオブジェクトに対する何らかの通信を表わしている。通信を受けた側のオブジェクトが何らかのリアクションを通信の発信元のオブジェクトに返すか返さないかに関わらず 1 本の矢印で表現されている。このため各矢印は「どちらが通信元であり、どちらが通信を受けるか」を表わす。各オブジェクトが持つインタフェースは 1 つのみであり、場合によって複数のインタフェースを使い分けるということはない。ただし、破線の矢印だけは例外であり、オブジェクトの生成という意味を持つ。

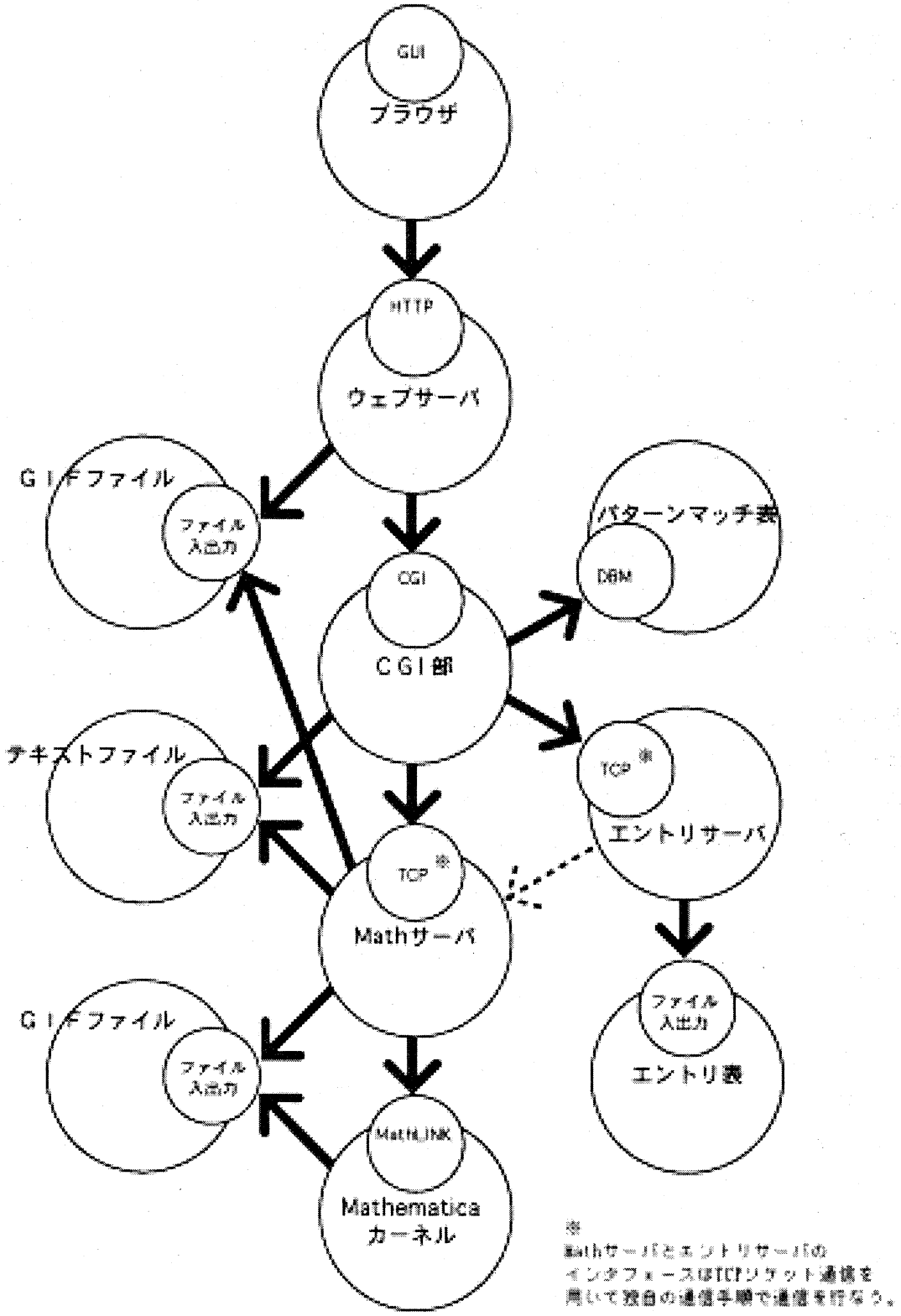

図 4: 各オブジェクトの関係

4 考察

非同期稼動の機能を付加したことにより、ユーザの思考を拘束する要素が一つ減らすことができた。「数式処理システムとは人間の思考を補助するツールである」という認識を出発点とし、人間の思考に重点をおいたシステムとしてミドルレギュレータを開発した。

一つの方向性として数式処理システムの高機能化は当然のことではある。しかし、人間の思考の補助という視点をおろそかにすると、高機能になればなるほどユーザにとって、さらに覚えなくてはいけない技能が増すことになる。

複数の数式処理システムを (ユーザにとっては) 同じに感じられるように使用するための調整役としてミドルレギュレータを作成した。

ミドルレギュレータは簡潔なインタフェースを持つオブジェクト群として開発されている。そのため複数の数式処理システムから、その数式処理エンジンの機能を得るためには、数式処理エンジンと通信するオブジェクトを交換するだけで良い。現在のミドルレギュレータにおいては、具体的には図 3 における Math サーバを交換することになる。

数式処理エンジンに対してのオブジェクトの交換と同様に、インタフェースに対してもオブジェクトの交換によって対応可能である。上と同様に図 3 における CGI 部が交換されるべきオブジェクトとなり、ウェブサーバとブラウザ以外のインタフェースに対応可能である。

ミドルレギュレータが一通り完成した後には、ユーザにとって負担の少ないユーザインタフェースについて研究を進めたい。

その際、open math などのオープンな規格を採用し、Mathematica 以外の数式処理エンジンを実際に利用可能にするべきであると考えている。

アプリケーションの呼び出しに CORBA の分散オブジェクト指向技術を導入することも検討中であり、このことについては「CORBA 版」と「非 CORBA 版 (現在のもの)」の両方向で試験的に作成することを予定している。

参 考 文 献

[1] DEGUCHI Hiroaki: The Integrated Use of Computer Algebra Systems across the Internet, *Proceedings of Asian Symposium on Computer Mathematics*, 1998, pp.101-105

[2] MathLink Reference Guide Version 2.2, Wolfram Research, Inc., 1993

[3] Todd Gayley: A MathLink Tutorial, *http://www.mathsource.com/cgi-bin/MathSource/Enhancements/MathLink /0206-693*

[4] Shishir Gundavaram: CGI プログラミング, 株式会社オライリージャパン, 1996